# Zadanie 0 – To co ostatnio (1 pkt)

Napisz funkcję

```
void minMaxRep(int a[], size_t size, int& mn, size_t& in,
                                              int& mx, size_t& ix);
```

która pobiera tablicę intów a, jej wymiar size oraz, przez referencje, cztery zmienne, do których wpisany ma być wynik działania funkcji: mn, in, mx i ix. Funkcja znajduje wartości najmniejszego i największego elementu tablicy i wpisuje je do zmiennych mn i mx, a do in i ix wpisuje, odpowiednio, liczbę wystąpień tej najmniejszej i największej wartości w całej tablicy.
Na przykład program:

```
#include <iostream>

void minMaxRep(int a[], size_t size,
int& mn, size_t& in, int&mx, size_t& ix)
{
        // ...
}

int main()
{
        using std::cout;
        int a[] = {2,3,4,2,7,4,7,2};
        size_t size = sizeof(a)/sizeof(*a);
        int mn, mx;
        size_t in, ix;
        minMaxRep(a,size,mn,in,mx,ix);
        cout << "Array: [ ";
        for (size_t i = 0; i < size; ++i)
                cout << a[i] << " ";
        cout << "]\n";
        cout << "Min = " << mn << " " << in << " times\n";
        cout << "Max = " << mx << " " << ix << " times\n";
}
```

powinien wydrukować:
**Array: [2,3,4,2,7,4,7,2]**
**Min = 2 3 times**
**Max = 7 2 times**
Uwaga: nie wolno stosować żadnych dodatkowych tablic ani kolekcji. Funkcja może przebiec w pętli po elementach tablicy tylko raz. Liczby w tablicy są zarówno ujemne jak i dodatnie.
Podpowiedź: Można wykorzystać kody z poprzednich zadań. Można założyć, że liczby w tablicy są z jakiegoś rozsądnego przedziału, np. $(-10^5, 10^5)$.

## Zadanie 1 – Odwrócenie tablicy (1 pkt)

Napisz program, który:

1. poprosi użytkownika o podanie n=10 liczb całkowitych i wczyta je do tablicy,
2. odwróci podaną tablicę, czyli z [1, 12, ... 5, 4] zrobi [4, 5, ... 12, 1] wykonując nie więcej niż n/2 zamian,
3. wyświetli odwróconą tablicę,
4. policzy dla odwróconej tablicy wariancję i wyświetli ją na standardowe wyjście.

Liczenie wariancji i odwracanie tablicy ma się odbywać za pomocą (co najmniej) dwóch funkcji, które będą przyjmowały argumenty typu int* (wskaźnik na int) oraz size_t (rozmiar tablicy). Program powinien umożliwiać łatwą zmianę n na inną liczbę, jeśli potrafisz to użyj stałej do zdefiniowania n. Program ma wyświetlać zrozumiałe i czytelne komunikaty.
Uwaga: Nie stosować kolekcji ani gotowych funkcji do operacji na tablicach. Żeby zdobyć pełną liczbę punktów należy ściśle spełnić podane warunki. Założyć, że użytkownik podaje 10 liczb całkowitych i nie trzeba tego sprawdzać.
Podpowiedź: n jest zdefiniowane w kodzie, nie jest pobierane od użytkownika.

Output:
Podaj 10 liczb odzielonych spacja i wcisnij ENTER
-1 0 1 -1 0 1 -1 0 1 0
0 1 0 -1 1 0 -1 1 0 -1
Wariancja wynosi: 0.6

## Zadanie 2 – ułamki egipskie (1,67 pkt)

Napisz funkcje, który wypisuje ułamki w postaci egipskiej, tj. jako sumę możliwie najmniejszej liczby ułamków o liczniku równym 1. Wykorzystaj poniższy szablon:

```
#include <iostream>

int main()
{
        int num[] = {3, 7, 0, 7, 123};
        int den[] = {4, 9, 8, 0, 43};
        for (int ii=0; ii<5; ii++)
        {
                int n = num[ii];
                int d = den[ii];
                std::cout << n << '/' << d << " -> ";
                egyptian(n, d);
                std::cout << std::endl;
        }
}
```

Output:
**3/4 -> 1/2 + 1/4**
**7/9 -> 1/2 + 1/4 + 1/36**
**0/8 -> 0**

**7/0 − > Denominator is 0!!!**
**123/43 − > 2 +1/2 + 1/3 + 1/37 + 1/9546**
Uwaga: Nie trzeba sprawdzać czy liczby są ujemne, zakładamy że są nieujemne.

Podpowiedź: Można to zrobić rekurencyjnie. Zastanów się jak można wykorzystać operator modulo, żeby znaleźć największy ułamek egipski nie mniejszy niż dany ułamek. Pamiętaj, żeby rozważyć różne przypadki liczb podawanych do funkcji, żeby rekurencja zawsze dobrze się kończyła.